11/1-23-90 JSO(1)

SANDIA REPORT

SAND90—0483 • UC—705
Unlimited Release
Printed May 1990

A Users' Manual for MCPRAM and for the Fuze Options in AMEER

R. A. LaFarge

DO NOT MICROFILM
COVER

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
US Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

SAND--90-0483

DE90 013598

SAND90-0483
Unlimited Release
Printed May 1990

A Users' Manual for MCPRAM and for the Fuze Options in AMEER

R. A. LaFarge
Aeroballistics Division
Sandia National Laboratories
Albuquerque, New Mexico 87185

Abstract

MCPRAM (Monte Carlo PReprocessor for AMEER), a computer program that uses Monte Carlo techniques to create an input file for the AMEER trajectory code, has been developed for the Sandia National Laboratories VAX and Cray computers. Users can select the number of trajectories to compute, which AMEER variables to investigate, and the type of probability distribution for each variable. Any legal AMEER input variable can be investigated anywhere in the input run stream with either a normal, uniform, or Rayleigh distribution. Users also have the option to use covariance matrices for the investigation of certain correlated variables such as booster pre-reentry errors and wind, axial force, and atmospheric models. In conjunction with MCPRAM, AMEER was modified to include the variables introduced by the covariance matrices and to include provisions for six types of fuze models. The new fuze models and the new AMEER variables are described in this report.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

3

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Contents

Nomenclature 6
1 Introduction 7
2 Single Variable Processing 9
3 Covariance Processing 13
3.1 A Normally Distributed Random Variable 13
3.2 Covariance Matrix 14
3.3 Pre-Reentry Covariance Matrix 17
3.4 Drag Covariance Matrix 19
3.5 Density Covariance Matrix 21
3.6 Wind Covariance Matrices 22
4 New Fuze Models in AMEER 23
4.1 Timer Fuze 23
4.2 G-Started Timer Fuze 27
4.3 Force-Balance Integrating Accelerometer (FBIA) Fuze 31
4.4 Path Length Fuze 33
4.5 Radar Fuze 36
4.6 Radar Updated Path Length (RUPL) Fuze 36
5 Conclusions 43
References 44

Tables

1 Aerodynamic Coefficient Multipliers 11
2 Mass Properties Multipliers 11
3 Propulsion System Multipliers 12
4 Wind Multipliers 12
5 Pre-Reentry AMEER Names 18
6 AMEER Names for the Timer Fuze Variables 24
7 AMEER Names for the G-Started Timer Fuze Variables 28
8 AMEER Names for the FBIA Fuze Variables 31
9 AMEER Names for the Path Length Fuze Variables 34
10 AMEER Names for the Radar Fuze Variables 36
11 AMEER Names for the RUPL Path Length Related Variables 38
12 AMEER Names for the RUPL Radar Related Variables 38

Figures

1 Normal Probability Density Function 15
2 Downrange Reference Frame 17
3 RUPL Fuze 37

Nomenclature

notational conventions

$(\vec{a})_W$	A vector in the W reference system. The vector can also be denoted as $(a_1, a_2, a_3)_W^T$ with magnitudes a_1, a_2, a_3 in the x, y, z directions respectively
$C_{\underline{x}}$	The covariance matrix defined by random vector $\underline{x}$
$E[x]$	The expected value of random variable x
$\overline{p}$	The mean value for the distribution of parameter p
$Pr[x]$	The probability of event x occurring
$\underline{x}$	A vector of N random variables. The vector can also be denoted as $(x_1, \ldots, x_N)^T$
δ_p	A randomly produced variation for parameter p
σ_p	The standard deviation for the distribution of parameter p
$\left[\mathbf{T}_{A/B}\right]$	denotes the linear transformation to reference frame A from reference frame B
P^T	The transpose of vector or matrix P
P^{-1}	The inverse of matrix P

symbols

CEP	circular error probability
C_{X_0}	axial force coefficient
h_{GD}	geodetic altitude
p	atmospheric pressure
P_k	probability of kill
$\dot{X}_W, \dot{Y}_W$	north/south and east/west wind velocities, respectively
γ_{GD}	vertical flight path angle
ρ	the atmospheric density
σ_{GD}	horizontal flight path angle
$\chi_{C_{X_0}}$	multiplier to the axial force coefficient

1 Introduction

MCPRAM (Monte Carlo PReprocessor for AMEER), a computer program that uses Monte Carlo techniques to create an input file for the AMEER[1,2] (Aero Mechanical Equation Evaluation Routines) six degree-of-freedom (6-DOF) trajectory code, has been developed for the Sandia National Laboratories (SNL) VAX and Cray computers. Users can select the number of trajectories to compute, which AMEER variables to investigate, and the type of probability distribution for each variable. Any legal input AMEER variable can be investigated anywhere in the input run stream with either a normal, uniform, or Rayleigh distribution. Users also have the option to use covariance matrices for the investigation of certain sets of correlated variables such as booster pre-reentry errors and wind, axial force, and atmospheric models. In conjunction with MCPRAM, AMEER was modified to include the variables introduced by the covariance matrices and to include models for six different types of fuzes. The new fuze models and the new AMEER variables are described in this report.

MCPRAM was written to take advantage of the merge case option[1] in AMEER in which initial and staging conditions are read for the computation of a nominal (or base) trajectory. Supplemental or merged trajectories are then computed by AMEER with only a few of the initial or staging conditions changed. This is exactly the type of structure that is used in Monte Carlo studies to investigate the effects of statistically varying parameters[3]. The nominal case represents the use of the mean values of the pertinent variables. Values for use in the merge cases are computed in the preprocessor through the use of random number generators having the specified probability distribution. Values must be supplied for the means and for either the standard deviations when the normal and Rayleigh distributions are specified or for an interval half length when the uniform distribution is specified.

Five different types of covariance matrices are predefined in MCPRAM and can be used to compute variations in seven sets of variables. The variables defined by the covariances were based on those that were modeled in MCTAFE (Monte Carlo Trajectory Analysis and Fuzing Environment) by R. C. Hartwig, 5166. Detailed explanations of a covariance matrix in general and how to specify each of the predefined covariances are included in Chapter 3; however, a brief description of each predefined covariance follows.

The pre-reentry covariance matrix (C_P) is a six by six matrix and is used to define the booster attributable errors in the position and velocity vectors of a reentry body. The errors are in position and velocity because they are defined at a reference time (the time at which the body on the nominal trajectory reenters the atmosphere). The six variables produced by the covariance analysis represent the XYZ error components for the velocity and the position vectors in the reference downrange/crossrange coordinate system. They are set to zero for the nominal case and are added to the initial velocity and position vectors in AMEER for the off-nominal trajectories. The errors can be in either English or metric units.

The drag covariance matrix ($C_{\underline{\chi}}$) introduces a table of $\chi_{C_{X_0}}$ as a function of altitude. The parameter $\chi_{C_{X_0}}$ is a multiplier to C_{X_0}, the axial force coefficient in AMEER. The size of the matrix is dependent on how many altitude points are investigated but cannot exceed 23 by 23. Each altitude can have a different $\chi_{C_{X_0}}$. MCPRAM sets up the table so that the multiplier is 1.0 for altitudes outside those in the table. Two covariances matrices ($C_{\underline{N}}, C_{\underline{E}}$) are used to investigate the winds. The north/south winds are computed from one covariance and the east/west from another. The covariance computations in MCPRAM produce tables of winds as functions of either altitude or atmospheric pressure, depending on a user-supplied flag. Again, the size of the covariance matrix is variable but cannot be greater than 23 by 23.

The fifth predefined covariance matrix ($C_{\underline{\rho}}$) is used to compute an atmospheric model for AMEER. The covariance computations produce a table of atmospheric density standard variations (σ_{ρ_i}) as a function of altitude. A table of mean values for the density ($\overline{\rho}_i$) at each altitude must be supplied. Also, values for the mean and the standard deviation of the atmospheric pressure ($\overline{p}_N$,σ_{p_N}) at the highest altitude (h_{GD_N}) defined by the covariance must be supplied. The *SITE* atmosphere option in AMEER (*IATMNM* = \$*SITE*\$) allows a user-supplied atmosphere to be specified. To satisfy the *SITE* atmosphere input requirements of AMEER, tables of density and pressure as functions of altitude must be computed in MCRPAM. The density table is computed by using the covariance information and a random number generator to compute random density variations (δ_{ρ_i}) as a function of altitude. The variations in density produced by the covariance analysis are added to the mean values supplied by the user at each altitude to compute the density table.

$$\rho_i = \overline{\rho}_i + \delta_{\rho_i} \qquad i = 1, \ldots, N \tag{1}$$

Then the standard deviation of the atmospheric pressure and a random number generator is used to compute a random variation in pressure at the highest altitude (δ_{p_N}). This is added to the pressure mean value to produce a random pressure value.

$$p_N = \overline{p}_N + \delta_{p_N} \tag{2}$$

The computed density and pressure values (ρ_N, p_N) are used as initial conditions to integrate the hydrostatic equation downwards in altitude to produce the atmospheric pressure at the next lowest altitude value ($(p)_{N-1}$). The process is repeated until the pressure at the lowest altitude is defined (p_1). The values for pressure and density as functions of altitude are then put into a format compatable to the AMEER *SITE* atmosphere option[1]. If the atmosphere defined by the covariance does not go to an altitude of 900 kft, the 1959 standard atmosphere is used in MCPRAM to complete the SITE atmosphere to that altitude.

MCPRAM was written with the intention that it could be used in conjunction with AMEER to study the effectiveness of certain types of fuzes; therefore, AMEER had to

be modified to include generic models for several types of fuzes. MCTAFE was again used as a source to determine what types of fuzes to include and how to model them for Monte Carlo studies. The generic fuze models include a timer fuze, a G-started timer fuze, a Force-Balance Integrating Accelerometer (FBIA) or velocity decrement fuze, a path length fuze, a radar fuze, and a Radar Updated Path Length (RUPL) fuze. The models include instrumentation error terms for timers, accelerometers, and radars that could be randomly generated by MCPRAM. Each fuze and its associated set of AMEER variables are explained in a later section.

2 Single Variable Processing

MCPRAM was written to be as little change to a standard AMEER input file as possible. In fact, most of the input lines are unprocessed except for a straight copy from the MCPRAM input file (*tp*5) to the AMEER input file (*tape*5). The only standard AMEER line that is processed is the $*RUN$ line. This the line used by AMEER to initiate its computational phase. Because of the way AMEER sets up merge cases, MCPRAM must keep track of the $*RUN$ lines to allow users to set up a Monte Carlo process anywhere in the input stream.

The preprocessor only checks to see if two things are present before it starts to compute the AMEER input file. It checks to see if $NRUNS$, the MCPRAM variable used to define the number of trajectories to be investigated, is supplied, and it checks to see if a $*END$ line has been included in *tp*5 since the $*END$ line is used by AMEER to signal the end of all computations. The number of trajectories defined by $NRUNS$ include the nominal trajectory as the first one. If either piece of information is missing, an error message is included in the AMEER input file and MCPRAM terminates. The designator $*MONTE$ (starting in the first column) is used by MCPRAM as a signal to start some kind of processing. All that has to be provided to initiate a Monte Carlo analysis of a single variable is the designator for the type of distribution the variable is to have (normal, uniform, or Rayleigh) and values for the mean and standard deviation. If the uniform distribution option is specified, the values for the midpoint and interval half length are given. Again, the $*MONTE$ designator is used to initiate the processing. The forms of the lines necessary to start the analysis of a variable or to define $NRUNS$ are described below.

$*MONTE$ $NRUNS$ [IVALUE]
$*MONTE$ [P-NAME] [DISTRIBUTION] $[\overline{P}]$ $[\sigma_P]$

An integer value is supplied for [IVALUE]; the AMEER parameter name is supplied for [P-NAME]; either $NORMAL$, $UNIFORM$, or $RAYLEIGH$ is supplied for [DISTRIBUTION]; the parameter's mean (or midpoint) value is supplied for $[\overline{P}]$; and the parameter's standard deviation (or interval half length) is supplied for $[\sigma_P]$. Random number generators from the IMSL[4] library are used in conjuntion with $[\overline{P}]$ and $[\sigma_P]$ to produce random values with the specified distribution properties. The line is read free

form except for *$MONTE$* starting in the first column. For example, if a user wants to use 1000 trajectories to investigate the effects of a normal distribution in altitude with a mean of 400,000 feet and a standard deviation of 500 feet, then the following two lines would be included in the MCPRAM input file.

```
*MONTE  NRUNS  1000
*MONTE  HGD8F  NORMAL  400000.0  500.0
```

Again, any variable can be investigated anywhere in the run stream including stage variables. If altitude were the first stage variable with the above properties, then the MCPRAM input file would contain:

```
NAMSTG(1)  $HGD8F$
*MONTE     VALSTG(1)  NORMAL  400000.0  500.0
```

The random number generators use double precision seeds to initiate the generating process. The MCPRAM variable *ISEED* designates the normal distribution seed; the variable *JSEED* designates the uniform distribution seed; and the variable *KSEED* designates the Rayleigh distribution seed. All have an initial default value of 585189929. Any other initial value for any of the seeds can be changed by a *$MONTE$* line similar to the one defining *NRUNS*. Since the seed value is changed each time a random number generator is called, MCPRAM specifies the value of the seeds at the start of each merge case computation on an AMEER comment line.

Multipliers to all the AMEER defined tables have been added to the list of legal AMEER names. This allows MCPRAM users to analyze statistical variations in aerodynamic coefficients, mass properties, propulsion systems properties, and winds since each multiplier can be used in a MCPRAM single variable Monte Carlo process. The default value for all multipliers is 1.0. Table 1 gives a list of the aerodynamic multipliers names and which AMEER aerodynamic coefficient table they multiply. Table 2 gives the names of the multiplers associated with mass properties. Table 3 gives the names of the multipliers associated with the nine propulsions systems that AMEER allows. Finally, the multipliers associated with the wind tables are given in Table 4.

As an example of how to use the multipliers, suppose a user wanted a Monte Carlo investigation of the normal force coefficient table (*TCZ0*) with a normal distribution and a standard deviation of 15 %, then the following line would be included in the MCPRAM input file:

```
*MONTE  XTCZ0  NORMAL  1.0  0.15
```

The MCPRAM single variable processing can also be used to investigate the effects of instrumentation errors in the the fuze models that will be discussed in Chapter 4. Instrumentation errors almost always have a normal distribution, a mean of 0.0 or 1.0, and are given in terms of one sigma values. For example, the AMEER name of the timer

AMEER name	Table Multiplied	AMEER name	Table Multiplied
XTCX0	TCX0	XTCXA	TCXA
XTCXA2	TCXA2	XTCXB	TCXB
XTCXB2	TCXB2	XTCXDQ	TCXDQ
XTCXDR	TCXDR	XTCY0	TCY0
XTCYB	TCYB	XTCYB2	TCYB2
XTCYB3	TCYB3	XTCYDR	TCYDR
XTCYP	TCYP	XTCYR	TCYR
XTCZ0	TCZ0	XTCZA	TCZA
XTCZA2	TCZA2	XTCZA3	TCZA3
XTCZDQ	TCZDQ	XTCZP	TCZP
XTCZQ	TCZQ	XTDELP	TDELP
XTDELQ	TDELQ	XTDELR	TDELR
XTXCP	TXCP		

Table 1. Aerodynamic Coefficient Multipliers

AMEER name	Table Multiplied	AMEER name	Table Multiplied
XTIXX	TIXX	XTIXX1	TIXX1
XTIXY	TIXY	XTIXY1	TIXY1
XTIXZ	TIXZ	XTIXZ1	TIXZ1
XTIYY	TIYY	XTIYY1	TIYY1
XTIYZ	TIYZ	XTIYZ1	TIYZ1
XTIZZ	TIZZ	XTIZZ1	TIZZ1
XTXCM	TXCM	XTYCM	TYCM
XTZCM	TZCM		

Table 2. Mass Properties Multipliers

AMEER name	Table Multiplied	AMEER name	Table Multiplied
XTMS11	TMS11	XTMS12	TMS12
XTMS21	TMS21	XTMS22	TMS22
XTMS31	TMS31	XTMS32	TMS32
XTMS41	TMS41	XTMS42	TMS42
XTMS51	TMS51	XTMS52	TMS52
XTMS61	TMS61	XTMS62	TMS62
XTMS71	TMS71	XTMS72	TMS72
XTMS81	TMS81	XTMS82	TMS82
XTMS91	TMS91	XTMS92	TMS92
XTVAC1	TVAC1	XTVAC2	TVAC2
XTVAC3	TVAC3	XTVAC4	TVAC4
XTVAC5	TVAC5	XTVAC6	TVAC6
XTVAC7	TVAC7	XTVAC8	TVAC8
XTVAC9	TVAC9		

Table 3. Propulsion System Multipliers

AMEER name	Table Multiplied
XTXWND	TXWIND
XTYWND	TYWIND
XTZWND	TZWIND

Table 4. Wind Multipliers

accuracy for the first timer fuze error is $CLKERR(1)$. The processing line, for a mean of zero and a one sigma value of 1500 parts per million (ppm), looks like:

```
*MONTE  CLKERR(1)  NORMAL  0.0  1500
```

3 Covariance Processing

Some time will be spent on the statistical definition of a covariance matrix so that a better understanding of the covariance processing in MCPRAM can be achieved. Basically, the covariance process was written to compute sets of random variables where the variables are correlated. That is, the value of one variable is dependent on the value of another. To construct such a set of random variables would be very difficult if not for the existence of a transformation to a set of independent random variables. Having random variables that are independent greatly simplifies their generation since all that is required is a call to an appropiate random number generator for each variable. As a result of having a set of independent random variables, the generation of the correlated variables is also simplified. Sections 3.1 and 3.2 offer a mathematical justification for the process used by MCPRAM to produce the correct set of correlated random variables. Although the analysis of the covariance manipulations should be understandable to most users, some background on random variables and linear algebra will be helpful. A discussion on the AMEER variables involved with each predefined covariance will then follow in the remainder of Chapter 3.

3.1 A Normally Distributed Random Variable

Suppose $\underline{x}$ is a vector of N random variables $x_1, \ldots, x_N$. The value taken by each random variable is the function of some probability distribution $\mathcal{F}(x)$ defined for random variable X by[5]

$$\mathcal{F}(x) = Pr(x \leq X) \tag{3}$$

The distribution density function $f(x)$ is defined by

$$f(x) = \frac{d\mathcal{F}(x)}{dx} \tag{4}$$

The expected value of a random variable ($E[X]$) is defined as the sum of all the values each weighted by the probability from which they were taken. In terms of the previous definitions, the expected value of random variable X with density function $f(x)$ is[6]

$$E[X] = \mu_x = \int_{-\infty}^{\infty} x f(x) dx \tag{5}$$

The expected value of any function $y(\underline{x})$ is defined as

$$E[y] = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} y f(x_1, \ldots, x_N) dx_x \ldots dx_N \tag{6}$$

The expected value is a linear function. That is, for constant c, random variable X, constant matrix C, and a matrix of random variables $\mathbf{X}$, the following relations are true.

$$E[cX] = cE[X] \tag{7}$$
$$E[C\mathbf{X}] = CE[\mathbf{X}] \tag{8}$$

The variance (σ_x^2), which is a measure of dispersion, can now be defined as:

$$\sigma_x^2 = E[(x - E[x])^2] = \int_{-\infty}^{\infty} (x - \mu_x)^2 f(x) dx \tag{9}$$

The standard deviation of a random variable is defined as the square root of the variance.

$$\sigma_x = \sqrt{\sigma_x^2} \tag{10}$$

For a normal distribution, the probability density function, with a standard deviation of σ_x and a mean of μ_x, is defined by

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma_x} \exp[-\frac{(x - \mu_x)^2}{2\sigma_x^2}] \tag{11}$$

A version (shown in Figure 1) of this function is used by the IMSL library. This version has a standard deviation of 1.0 and a mean of 0.0. The advantage of using this version of the density function is that the random number generated from this distribution can then be multiplied by any standard deviation σ_x and added to any mean μ_x to produce a new random variable whose distribution has a mean of μ_x and a standard deviation of σ_x.

The density functions for the uniform and Rayleigh probability distribution have slightly different forms. They are not shown because only a normal distribution is used for the covariance analysis.

3.2 Covariance Matrix

Suppose, for the sake of simplicity, that only two random variables (x and y) exist. The analysis can always be expanded to a larger set of random variables. Then a covariance σ_{xy} can be defined by

$$\sigma_{xy} = E[(x - E[x])(y - E(y)] = E[xy] - \mu_x \mu_y \tag{12}$$

If x and y are independent, then

$$E[xy] \equiv E[x]E[y] \tag{13}$$

and

$$\sigma_{xy} = 0 \tag{14}$$

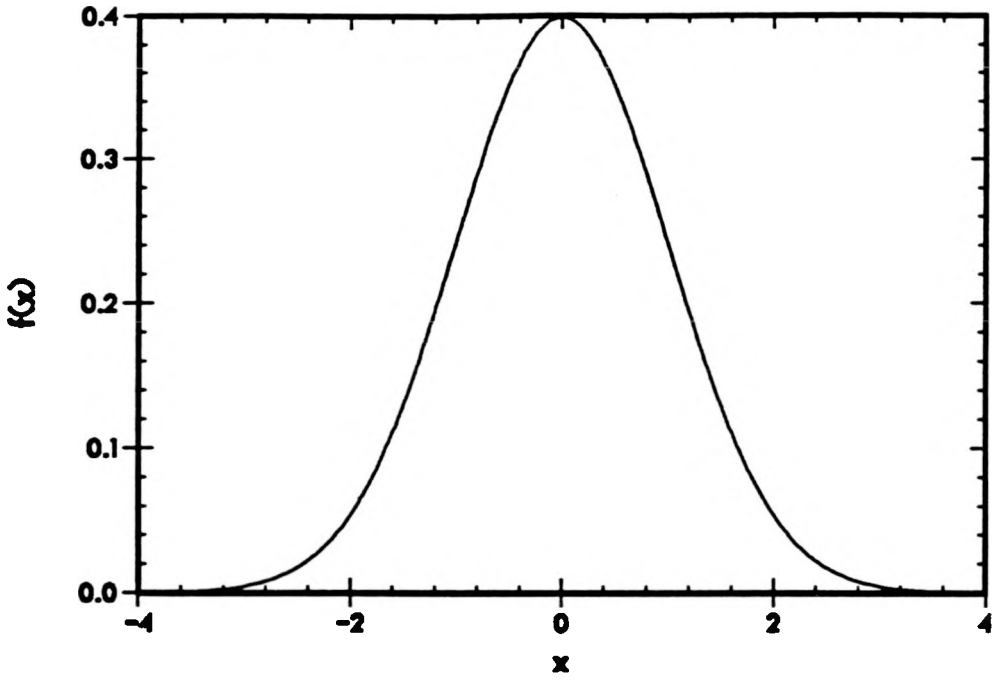

Figure 1. Normal Probability Density Function

If x and y are then assumed to be the two components of vector $\underline{v}$, then a covariance matrix on $\underline{v}$ can be defined by

$$C_{\underline{v}} = E[(\underline{v})(\underline{v}^T)] = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} \tag{15}$$

Note that $C_{\underline{v}}$ is symmetric. For a vector $\underline{w}$ made up of independent random variables c and d, the covariance of $\underline{w}$ is

$$C_{\underline{w}} = \begin{bmatrix} \sigma_c^2 & 0 \\ 0 & \sigma_d^2 \end{bmatrix} \tag{16}$$

since $\sigma_{cd} = 0$.

If MCPRAM can somehow work with $\underline{w}$, then d could be generated independent of the value of c. This greatly simplifies the work that has to be done by MCPRAM.

Now suppose that MCPRAM has been given $C_{\underline{v}}$, the covariance of random vector $\underline{v}$ with $\sigma_{xy} \neq 0$. MCPRAM needs to find a coordinate transformation matrix B such that

$$\underline{w} = B\underline{v} \tag{17}$$

A proof of the existence of such a transformation may be found in Reference 6, which proves that such a transformation to a vector of independent random variables is always

possible. If zero means are assumed, the covariance of $\underline{w}$ can be defined by

$$C_{\underline{w}} = E[(B\underline{v})(B\underline{v})^T] = BC_{\underline{v}}B^{-1} \tag{18}$$

since B is a constant transformation matrix. Note that $C_{\underline{w}}$ is a diagonal matrix. Equation 18 has the exact form of the diagonalization of a symmetric positive definite matrix. That is, if A is a symmetric matrix, and Λ is a diagonal matrix of the eigenvalues of A, then

$$\Lambda = P^{-1}AP \tag{19}$$

where P is a matrix whose columns are the eigenvectors of A[7]. If one assumes that A and $C_{\underline{v}}$ are the same and that the eigenvalues of $C_{\underline{v}}$ are λ_1 and λ_2, then the transformation matrix B can be defined such that

$$\sigma_c = \sqrt{\lambda_1} \tag{20}$$

$$\sigma_d = \sqrt{\lambda_2} \tag{21}$$

$$\underline{v} = B^{-1}\underline{w} = P\underline{w} \tag{22}$$

The implication of defining this transformation is that random numbers c and d can now be generated independent of each other using σ_c and σ_d and a random number generator. The normal random number generator in the IMSL math library is used in MCPRAM since Hartwig found that the random numbers were better distributed than those generated by the SLATEC routines. All that would be necessary is some algorithm for computing the eigenvalues (λ_1 and λ_2) and the eigenvector matrix (P) of $C_{\underline{v}}$. Although such algorithms exist in various math libraries, the IMSL routines were used since IMSL was already being used to generate the random numbers.

To summarize the procedure used by MCPRAM to generate correlated random numbers when a covariance matrix $C_{\underline{v}}$ is specified:

1. Use IMSL to compute eigenvalues and the eigenvector matrix of $C_{\underline{v}}$.

2. Use the standard deviations defined by the square roots of the eigenvalues and a normalized random number generator to compute $\underline{w}$, a vector of independent random numbers.

3. Use the eigenvector matrix to tranform $\underline{w}$ to $\underline{v}$, the vector of correlated random numbers.

Step 1 is only performed once. Steps 2 and 3 are performed each time a merge trajectory is computed. Since each covariance matrix must be symmetric, MCPRAM generates an errror message if it is not.

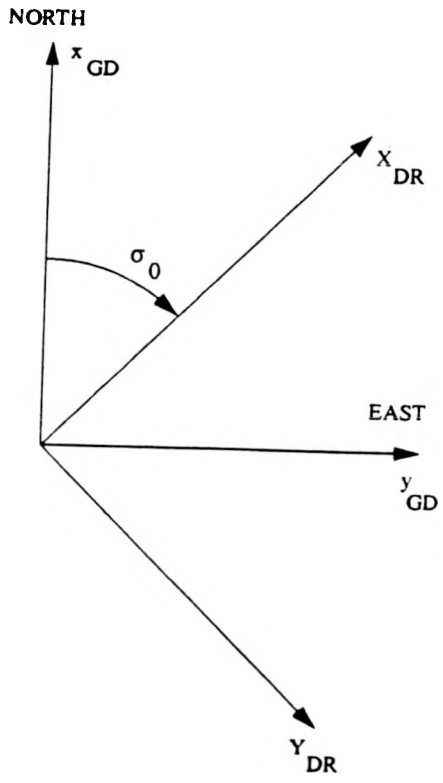

Figure 2. Downrange Reference Frame

3.3 Pre-Reentry Covariance Matrix

The pre-reentry covariance matrix $C_{\underline{P}}$ is used to describe the booster induced errors to the velocity and position vectors at reentry, that is, at the trajectory's initial conditions. Modifications were made in AMEER's initialization phase (as defined in Reference 2) to account for the introduction of these errors. The random vector ($\underline{P}$) defining the covariance is composed of the six XYZ errors to velocity and position defined in the reference downrange coordinate system and is given by

$$\underline{P} = \begin{pmatrix} \delta_{\dot{X}} \\ \delta_{\dot{Y}} \\ \delta_{\dot{Z}} \\ \delta_X \\ \delta_Y \\ \delta_Z \end{pmatrix}_{DR} \tag{23}$$

The reference downrange coordinate system $(X, Y, Z)_{DR}$ is aligned with the local geodetic reference frame $(x, y, z)_{GD}$ except for a rotation in the xy plane of σ_0, the reference horizontal flight path angle (AMEER name of $SIGZD$). The relative orientation of the x and y axes is illustrated by Figure 2.

MCPRAM computes the pre-reentry errors for each off-nominal trajectory. The

AMEER name	Pre-Reentry Error	AMEER name	Pre-Reentry Error
DLDRF1	$\delta_{\dot{X}}$	DELDRF	δ_X
DLCRF1	$\delta_{\dot{Y}}$	DELCRF	δ_Y
DELHF1	$\delta_{\dot{Z}}$	DELHF	δ_Z

Table 5. Pre-Reentry AMEER Names

English units AMEER names of the pre-reentry errors are given in Table 5 (metric unit names replace the F with an M).

AMEER Modification

For AMEER to make use of the pre-reentry errors generated by MCPRAM, certian modifications were made in AMEER's initialization routine. To initiate the AMEER processing of the pre-reentry errors, the new AMEER flag $IXPRE$ must be set to 1. Also the geodetic input option must be used. $IXPRE$ initiates the computation of the downrange to geodetic transformation ($[T_{GD/DR}]$) right after the computation of the local geodetic velocities. The transformation is defined by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{GD} = \begin{pmatrix} \cos\sigma_0 & -\sin\sigma_0 & 0 \\ \sin\sigma_0 & \cos\sigma_0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{DR} \tag{24}$$

The velocity error vector ($\vec{\delta v}_{DR}$) is then transpose to the geodetic reference frame and added to the nominal geodetic velocity vector.

$$\vec{v}_{GD} = \vec{v}_{GD} + [T_{GD/DR}]\vec{\delta v}_{DR} \tag{25}$$

The resulting vertical and horizontal flight path angles are computed from the updated geodetic velocities in the normal manner.

$$\sigma_{GD} = \arctan\left(\frac{\dot{y}_{GD}}{\dot{x}_{GD}}\right) \tag{26}$$

$$\gamma_{GD} = \arcsin\left(\frac{-\dot{z}_{GD}}{V_{GD}}\right) \tag{27}$$

The new velocities and flight path angles are treated in the same manner as prescribed in Reference 2 (Section 3).

The position errors are treated slightly different. A transformation from the downrange reference frame to the local geocentric reference frame $(x, y, z)_{lgc}$ is computed just before the inertial position vector($\vec{R}_I$) is computed. The transformation is given by

$$[T_{lgc/DR}] = [T_{lgc/GD}][T_{GD/DR}] \tag{28}$$

with the geodetic to local geocentric transformation as defined in Equation 23 in Reference 2. The position error vector ($\delta\vec{R}_{DR}$) is transpose to the local geocentric reference frame. This vector is added to the local geocentric position vector ($\vec{R}_{lgc}$) just before it is used to compute the inertial position vector.

$$\vec{R}_I = [T_{I/lgc}](\vec{R}_{lgc} + [T_{lgc/DR}]\delta\vec{R}_{DR}) \tag{29}$$

MCPRAM Input

To initiate the pre-reentry covariance processing in MCPRAM, a user must include a *COVARIANCE* line in the MCPRAM input file. It has the following form:

**MONTE COVARIANCE PRE* 6 *NEG METRIC* [SCALE]

where [SCALE] is an optional scale factor to the standard deviations. Not including any value forces a default to a value of 1.0. *NEG* is an optional flag to tell MCPRAM to use the absolute value of any negative eigenvalue rather then setting the eigenvalue to zero. There is no formal justification for this option other than it was requested by Hartwig. *METRIC* is an optional designator that metric units are being used. The pre-reentry covariance is specified right after this line. It is read in a free format per line. That is, six values are read before the next line can be specified. This pre-reentry is the easiest of the predefined covariance matrices to define for processing.

3.4 Drag Covariance Matrix

The drag covariance matrix is used to determine a table of axial force coefficient multipliers as a function of altitude. The name *TXTCX*0 was added to the list of legal AMEER table names to accommodate the results of the covariance matrix processing. The table follows the same rules that other AMEER tables follow as described in Reference 1. The result of an interpolation on this table is a value for the AMEER parameter *XTCX*0, which has been previously described as a multiplier to the AMEER axial force coefficient. The computed random multipliers have a zero mean and are in units of percent; therefore, they are divided by 100.0 and added to 1.0 in MCPRAM to have the proper range. To initiate the drag covariance processing, a user must include the following line in the MCPRAM input file.

**MONTE COVARIANCE DRAG* [NVALUE] *NEG* [SCALE]

where [*NVALUE*] is the dimension of the table, and *NEG* and [*SCALE*] are as before. The table can contain up to 23 entries. Tables are computed for only the off-nominal trajectories. MCPRAM sets up the table so that the extrapolated values of $\chi_{c_{x_0}}$ are 1.0. The drag covariance matrix is specified after the drag covariance line. Again, it is read in a free format per line. The tabular values for altitude (ENGLISH units) are read next in a free format. A typical example for an 18 by 18 matrix follows. The last three lines represent the altitude table.

*MONTE COVARIANCE DRAG 18 NEG

22.666279040	10.667540105	15.665680281	14.111049818	7.778145804	5.778259624
5.444009656	4.999786028	2.999915812	2.555282097	-1.222276277	-1.666844521
-.999884528	-.444656241	-.222287448	-.999884528	-.666490920	.222052128
10.667540105	18.278190090	12.289061289	7.722888581	2.499852074	-2.554965220
-6.224128859	-2.222978042	1.500044669	6.777647797	6.444429024	5.500501451
2.277880866	-2.111175205	1.610954417	2.277880866	5.778182511	2.500160662
15.665680281	12.289061289	16.611220490	12.500864256	8.288742021	5.777929227
5.556122202	6.222451215	6.499899216	5.222211116	.222542920	-1.822288009
-1.944207695	-2.444227607	.055620452	-1.944207695	2.221980222	4.166748122
14.111049818	7.722888581	12.500864256	11.722406440	7.722570168	6.666660182
7.111125949	6.999708991	5.822566854	2.222471929	-1.555490887	-2.166822128
-2.288982019	-2.555716728	-.288825820	-2.288982019	.444522257	2.822257952
7.778145804	2.499852074	8.288742021	7.722570168	5.822191040	6.222105228
7.221868245	6.666849657	4.822281271	1.666484522	-2.444527000	-2.822481181
-2.611020259	-2.111212509	-.611076272	-2.611020259	-.444405688	2.166714901
5.778259624	-2.554965220	5.777929227	6.666660182	6.222105228	8.888745960
11.111152020	9.222857600	5.222402472	-.444251126	-5.222151277	-6.666852244
-2.999702848	-1.777847280	-1.222161222	-2.999702848	-2.666915627	1.222088528
5.444009656	-6.224128859	5.556122202	7.111125949	7.221868245	11.111152020
14.111292250	11.666542629	6.222492710	-1.222444262	-7.110877118	-8.667025950
-5.110778655	-1.999992155	-1.777550621	-5.110778655	-2.777862289	1.222295782
4.999786028	-2.222978042	6.222451215	6.999708991	6.666849657	9.222857600
11.666542629	10.000141290	5.999802444	0.000000000	-5.222408688	-7.000175684
-4.222226578	-2.222165495	-1.222225680	-4.222226578	-2.222297015	1.999965012
2.999915812	1.500044669	6.499899216	5.822566854	4.822281271	5.222402472
6.222492710	5.999802444	4.499912690	1.666485898	-1.999924148	-2.499822987
-2.500051497	-2.222285752	-.499926105	-2.500051497	0.000000000	2.500062220
2.555282097	6.777647797	5.222211116	2.222471929	1.666484522	-.444251126
-1.222444262	0.000000000	1.666485898	2.110990440	2.222141622	1.222182270
.222222622	-1.666584459	.555482706	.222222622	2.555502242	2.222242255
-1.222276277	6.444429024	.222542920	-1.555490887	-2.444527000	-5.222151277
-7.110877118	-5.222408688	-1.999924148	2.222141622	4.444507240	4.666782789
2.444228998	0.000000000	1.111021400	2.444228998	2.111062687	.666655004
-1.666844521	5.500501451	-1.822288009	-2.166822128	-2.822481181	-6.666852244
-8.667025950	-7.000175684	-2.499822987	1.222182270	4.666782789	5.499962040
2.166482224	1.000022852	1.166585266	2.166482224	2.666545720	-.499827822
-.999884528	2.277880866	-1.944207695	-2.288982019	-2.611020259	-2.999702848
-5.110778655	-4.222226578	-2.500051497	.222222622	2.444228998	2.166482224
1.944251260	.999994657	.611026777	1.944251260	1.111126622	-.822271248

```
-.444656241  -2.111175205  -2.444227607  -2.555716728  -2.111212509  -1.777847280
-1.999992155 -2.222165495  -2.222285752  -1.666584459   0.000000000   1.000022852
 .999994657   1.666681000   0.000000000   .999994657  -1.000114669  -1.999967858
-.222287448   1.610954417   .055620452  -.288825820   -.611076272   -1.222161222
-1.777550621 -1.222225680  -.499926105   .555482706   1.111021400   1.166585266
 .611026777   0.000000000   .277729000   .611026777   .777691887    .166647940
-.999884528   2.277880866  -1.944207695  -2.288982019  -2.611020259  -2.999702848
-5.110778655 -4.222226578  -2.500051497   .222222622   2.444228998   2.166482224
 1.944251260   .999994657   .611026777   1.944251260   1.111126622  -.822271248
-.666490920   5.778182511   2.221980222   .444522257  -.444405688   -2.666915627
-2.777862289 -2.222297015   0.000000000   2.555502242   2.111062687   2.666545720
 1.111126622 -1.000114669   .777691887   1.111126622   2.777888890   1.666776252
 .222052128   2.500160662   4.166748122   2.822257952   2.166714901   1.222088528
 1.222295782   1.999965012   2.500062220   2.222242255   .666655004  -.499827822
-.822271248  -1.999967858   .166647940  -.822271248   1.666776252   2.499877210
   0.0    5000.0   10000.0   15000.0   20000.0   25000.0   30000.0   35000.0
40000.0   45000.0   50000.0   55000.0   60000.0   65000.0   70000.0   75000.0
80000.0   85000.0
```

Note that since only six values per line are used to specify the drag covariance matrix, each row of the 18 by 18 matrix takes three lines of input.

3.5 Density Covariance Matrix

The density covariance matrix is used to compute variations in density ($(\delta\rho)_i$) from a mean density profile ($\overline{\rho}_i$) as a function of altitude. These variations and a random value in pressure eventually produce a random atmospheric model for AMEER. The procedure has been described in Chapter 1. To flag MCPRAM to initiate the density covariance processing, the following line must be included in the MCPRAM input file.

**MONTE COVARIANCE DENSITY* [NVALUE] *NEG* [SCALE]

The parameter [NVALUE] again denotes the size of the density table (≤ 23), and *NEG* and [SCALE] are as on the previous *COVARIANCE* lines. MCPRAM next looks for an auxiliary line that specifies the mean and standard deviation for pressure ($\overline{p}_N, \sigma_{p_N}$) at the maximum altitude. It also looks for some additional optional scale factors. The line has the following form. Because of spacing limitations, it is shown on two lines.

PM(N) [PVALUE] *SIGMAP* [SVALUE] *SCALEP* [SPVALUE]
SCALEH [SHVALUE] *SCALER* [SRVALUE] *SCALEB* [SBVALUE]

The value for $\overline{p}_N$ is substituted for [PVALUE]; the value for σ_{p_N} is substituted for [SVALUE]. *SCALEP* denotes an optional scale factor to the pressure; *SCALEH* denotes an optional altitude scale factor; *SCALER* denotes an optional scale factor to the mean density table; and *SCALEB* denotes an optional scale factor to the pressure standard

deviation. The designators $PM(N)$ and $SIGMAP$ with their corresponding values must be included on this line, or an error message will be printed on *tape*5 and MCPRAM will terminate. The scale factors are included as optional parameters so that the covariance can be specified in arbitrary units and converted to English units through them. The values for the scale factors will default to 1.0 if not included on this line. A typical example of the $COVARIANCE$ line and its auxiliary line follow.

```
*MONTE COVARIANCE DENSITY 20 NEG .00000194030
SCALEB 2.8854000 SIGMAP .366 PM(N) 3.90 SCALEP 2.0885400 SCALER .0019403
```

The density covariance matrix is specified next in a free format per line similar to the previous example. After the covariance matrix, the [NVALUE] tabular values of altitude are specified in a free format. Finally, the tabular values of $\overline{\rho}_i$ are read, also in a free format. For the nominal atmosphere, MCPRAM uses the mean values of the density table and the mean value of pressure at the maximum altitude to construct the model. MCPRAM then uses the 1959 standard atmosphere to complete the model to 900 kft.

3.6 Wind Covariance Matrices

Two covariance matrices are used to compute wind variations. One ($C_{\underline{N}}$) produces the north/south wind variations, and the other ($C_{\underline{E}}$) produces the east/west variations. AMEER tables of wind velocities as functions of either altitude or atmospheric pressure are produced by MCPRAM. An auxiliary line is used to specify the desired independent variable. The $COVARIANCE$ line has the form:

$$*MONTE \quad COVARIANCE \quad \text{[C-NAME]} \quad \text{[NVALUE]} \quad NEG \quad \text{[SCALE]}$$

where [C-NAME] is replaced by $NSWINDS$ for the north/south winds or by $EWWINDS$ for the east/west winds. The auxiliary line has the form:

$$INWIND \quad \text{[IVALUE]} \quad SCALEA \quad \text{[SVALUE]}$$

where [IVALUE] is replaced by one for altitude as the independent variable or by two for pressure as the independent variable. $SCALEA$ designates an optional scale factor to the independent variable so that it is in English units. The covariance matrix is placed right after the auxiliary line. The tabular values for the independent variable are placed right after the covariance matrix. The same rules as before for reading the matrix and the table apply. Wind tables are computed only for the off-nominal trajectories. The tables are set up such that extrapolated values are zero.

This completes the discussion on MCPRAM. A detailed explanation on the new fuze models in AMEER follows.

4 New Fuze Models in AMEER

Interest in evaluating the effectiveness of a missile system has existed at SNL for some time. The effectiveness of a missile system is determined as much by the guidance system errors and fuzing capability as by the CEP or yield. Although empirical expressions for P_k exist, they make simplifying assumptions about burst distributions that are not valid for some fuze options. For this reason, there is a need for a tool that computes burst distributions from complete trajectories. The tool, in this case the AMEER trajectory code, should have provisions for modeling both guidance and fuzing errors. The provision for modeling guidance system or pre-reentry errors has been previously discussed. This chapter will cover the new fuze models incorporated into the AMEER trajectory code. The models were written so that MCPRAM and AMEER would be used together to provide a means of evaluating missile system effectiveness from a Monte Carlo study. The Monte Carlo method was chosen so that comparisons with a previous work could be made. AMEER itself only computes the burst point locations for each fuze in each trajectory. A separate computer program SCORES is needed to actually determine the fuzing effectiveness.

Six types of fuze models with their associated instrumentation errors currently exist in AMEER. The six fuze types include timer, g-started timer, FBIA, path length, radar, and RUPL. Each type of fuze can have up to five different versions running simultaneously in AMEER. Each version can have up to ten burst locations to investigate per trajectory. The new AMEER variable, $IXFUZE$, is set to 1 to initialize the fuze option. The downrange, crossrange, and altitude values of each burst point in the reference downrange coordinate system (Section 3.3) are printed for each fuze version on AMEER output file, *tape*25. Another AMEER flag, $IXMNTE$, can be set to 1 to suppress printing of the other output files to avoid overwhelming the computer disks with information. Since the purpose of this modification to AMEER is to model the the fuzing errors contributions to burst location, each error will be described in detail. Hartwig's $MCTAFE$ code was used as a reference for each error model.

4.1 Timer Fuze

The timer fuze model is the easiest to understand. It's only instrumentation error is clock accuracy. If t_{out_i} represents the modeled output of timer i, Δt_{ref_i} is the difference in time between some fiducial time (the time at which the fuze takes over internally from the missile system) and the nominal reentry time, t is the time from reentry, and ϵ_i is the timer accuracy in parts per million (ppm); then the timer output is modeled as:

$$t_{err_i} = 1,000,000 \times \epsilon_i \tag{30}$$

$$t_{out_i} = (t_{err_i})(\Delta t_{ref_i}) + (1 + t_{err_i})t \tag{31}$$

with

$$i = 1, \ldots, N_{timer} \tag{32}$$

AMEER name	Timer variable	AMEER name	Timer variable	AMEER name	Timer variable
TIMVL1	t_{out_1}	CLKPPM(1)	ϵ_1	CLKSRT(1)	Δt_{ref_1}
TIMVL2	t_{out_2}	CLKPPM(2)	ϵ_2	CLKSRT(2)	Δt_{ref_2}
TIMVL3	t_{out_3}	CLKPPM(3)	ϵ_3	CLKSRT(3)	Δt_{ref_3}
TIMVL4	t_{out_4}	CLKPPM(4)	ϵ_4	CLKSRT(4)	Δt_{ref_4}
TIMVL5	t_{out_5}	CLKPPM(5)	ϵ_5	CLKSRT(5)	Δt_{ref_5}

Table 6. AMEER Names for the Timer Fuze Variables

$$N_{timer} \leq 5 \tag{33}$$

There is a problem with how the timer fuze is modeled in that fuzes settings are really determined by the desired burst altitude and not by a countdown from reentry. Therefore, a preliminary AMEER trajectory (with all nominal values) is necessary to determine the values for time at the correct altitudes (these are called the timer preset values). Since the timer errors are zero for the nominal trajectory, the output of each timer (t_{out_i}) defaults to the trajectory time. The user can create any output file with time as a variable with staging occurring at whatever altitudes need investigating. Once the nominal presets are determined, MCPRAM is used not only for generating pre-reentry errors and the off-nominal atmospheric, wind, and drag models, but also for generating a random value for each timer error. Usually, timer accuracy is given in terms of a one sigma value. The best way to illustrate the proper procedure to follow is to give an example. Note that the values used in all the examples have no meaning other than to illustrate the use of the fuze algorithms. The AMEER names that are necessary to define a typical input file must first be presented.

The AMEER names for the timer fuze variables are shown in Table 6. Additionally, the AMEER variable $NTIMER$ defines the number of timer fuzes and $NUMFUZ$ defines the total number of all fuzes. For output purposes, the AMEER array $NAMFUZ(I)$ defines a six character designator for the burst point of each fuze preset. Setting $IXFUZE$ to zero makes AMEER print out all the values for the fuze parameters as well as the burst point information on *tape*25. Setting $IXFUZE$ to one allows AMEER to use *tape*25 as just the burst point output file. For each trajectory, the downrange, crossrange and altitude values of the burst point for each preset of each fuze is printed next to the $NAMFUZ$ designator. Before the burst point information is printed, a header line identifying which MCPRAM trajectory is being computed is printed. This is to assist the user in identifying the correct set of initial conditions whenever a problem occurs. The AMEER array $PRTMVL(I,J)$ is used to define the preset value J for timer I. The array $NTPRST(I)$ is used to define the number of preset values for timer I. Because more than one fuze can go off at any one stage, the AMEER variable $FUZERR$ is used

to check if any fuze output is within $FUZERR$ of its preset value.

Before an illustrating example can be given, an option that is useful to the post-processor SCORES should be explained. Because SCORES may have to interpolate in altitude between burst points, an option to print downrange and crossrange information at specific altitudes was included in the AMEER modifications. This altitude option is treated as a fuze model internally and is always the first stage to be specified in the input file. The AMEER variable $NMTALT$ is used to describe number of altitudes to be printed ($NMTALT \leq 10$). The array $HTRGTM(I)$ is used the specify the desired altitudes (in metric units) in descending order. $HTRGTM(2)$ has an additional role to play in that it is the altitude of the actual target (h_{target}). Its value is subtracted from the AMEER computed altitude to get an altitude to target value for radar modeling purposes. $HTRGTM(1)$ also has a dual role. It is used as the cutoff altitude. No trajectory information is computed for altitudes below its value because the burst points would be considered misses anyway.

An example best illustrates the use of this array. Let's suppose that a user was interested in making a Monte Carlo study of two timer fuzes on the same missile system. The first fuze has a timer accuracy of 1000 ppm and the second has one of 1500 ppm. Futhermore, the target is at 5000 meters and Δt_{ref} has been given as 800 seconds. Burst points below 2000 meters are not to be computed and altitudes of 1000 and 4000 meters above the target are needed for interpolation. Two burst point altitudes, 3000 and 2000 meters above target, are to be investigated for the first fuze, and three burst point altitudes (3000, 2500, and 1500 meters above target) are to be investigated for the second. A preliminary trajectory calculation has shown that for timer 1, the altitudes in question occur at 40 and 41 seconds after reentry, and for timer 2, they occur at 40.0, 40.7, and 41.3 seconds. All the body's mass properties, initial conditions, aerodynamic coefficients, and the covariance matrices have already been defined on the MCPRAM input file *tp5*. Then the following lines must be added to *tp5* to properly model the altitude option with the two timer fuzes. (Anything after // is considered a comments by AMEER.)

```
IXFUZE 1 // fuze flag
IXMNTE 1 // output suppress flag
NTIMER 2 // number of timer fuzes
NUMFUZ 9 // total number of presets for all fuzes
FUZERR 1.0E-04 // fuze preset error tolerance
NMTALT 4 // number of target altitudes
HTRGTM(1) 9000.0 //first altitude
HTRGTM(2) 6000.0 //second altitude
HTRGTM(3) 5000.0 //target altitude
HTRGTM(4) 2000.0 //cutoff altitude
NTPRST(1) 2 // number of presets for timer 1
NTPRST(2) 3 // number of presets for timer 2
```

```
CLKSRT(1) 800.0 // Delta time_ref for timer 1
CLKSRT(2) 800.0 // Delta time_ref for timer 2
*MONTE CLKPPM(1) NORMAL 0.0 1000.0
*MONTE CLKPPM(2) NORMAL 0.0 1500.0
PRTMVL(1,1) 40.0 // timer 1 preset 1
PRTMVL(1,2) 41.0 // timer 1 preset 2
PRTMVL(2,1) 40.0 // timer 2 preset 1
PRTMVL(2,2) 40.7 // timer 2 preset 2
PRTMVL(2,3) 41.3 // timer 2 preset 3
NAMFUZ(1) $HCTOFF$ //cutoff altitude
NAMFUZ(2) $HTRGT$ //target altitude
NAMFUZ(3) $H6000$ //6000. meters altitude
NAMFUZ(4) $H9000$ //9000. meters altitude
NAMFUZ(5) $TMR1P1$ //timer 1 preset 1
NAMFUZ(6) $TMR1P2$ //timer 1 preset 2
NAMFUZ(7) $TMR2P1$ //timer 2 preset 1
NAMFUZ(8) $TMR2P2$ //timer 2 preset 2
NAMFUZ(9) $TMR2P3$ //timer 2 preset 3
NAMSTG(1) $HGD8M$ //stage on first altitude
VALSTG(1) 9000.0
NAMSTG(2) $TIMVL1$ //stage on first preset for timer 1
VALSTG(2) 40.0
NAMSTG(3) $TIMVL2$ //stage on first preset for timer 2
VALSTG(3) 40.0
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*END
```

Note that there are nine $*RUN$ lines signifying the nine AMEER stages for the altitudes of interest. This structure allows for the uncertainty in the relative behavior among the different fuzes. The user does not need to know if preset 1 of timer 1 goes off before preset 1 (or 2) of timer 2. All that matters is that nine pieces of information should be printed per trajectory. Also, there is a duplication of some information simply because some basic AMEER subroutines would have had to be rewritten to avoid the duplication. Another feature to notice is that the output of each fuze is a staged variable

(*HGD8M*, *TIMVL1*, and *TIMVL2*) and the first preset for each fuze is the staged value. Once that first preset value is staged, AMEER automatically puts in the next preset value. That is, once the trajectory reaches a timer 1 output value of 40.0 seconds, an output value of 41.0 seconds is used for the next stage on that fuze. **Notice should be taken that, because of the two features just described, all fuze related variables should be specified only in metric units even if the English units option is being used.** The information printed on *tape25* for the first two trajectories should look like (metric option):

```
MONTE  CARLO RUN          1
            HCTOFF  2.8487000e+05   2.5700000e+01   2.0000000e+03
            HTRGET  2.7840000e+05   2.5400000e+01   5.0000000e+04
              H6000 2.7630000e+05   2.5300000e+01   6.0000000e+04
              H9000 2.6985000e+05   2.4000000e+01   9.0000000e+03
            TMR1P1  2.7200000e+05   2.5000000e+01   8.0000000e+03
            TMR1P2  2.7410000e+05   2.5200000e+01   7.0000000e+03
            TMR2P1  2.7200000e+05   2.5000000e+01   8.0000000e+03
            TMR2P2  2.7300000e+05   2.5100000e+01   7.5000000e+03
            TMR2P3  2.7520000e+05   2.5250000e+01   6.5000000e+03
MONTE  CARLO RUN          2
            HCTOFF  2.8518000e+05  -2.5700000e+01   2.0000000e+03
            HTRGET  2.8145000e+05  -2.5400000e+01   5.0000000e+04
              H6000 2.7852100e+05  -2.5300000e+01   6.0000000e+04
              H9000 2.9887200e+05  -2.4000000e+01   9.0000000e+03
            TMR1P1  2.7500000e+05  -2.5000000e+01   8.0100000e+03
            TMR1P2  2.7710000e+05  -2.5200000e+01   7.0110000e+03
            TMR2P1  2.7510000e+05  -2.5000000e+01   8.0090000e+03
            TMR2P2  2.7610000e+05  -2.5100000e+01   7.5110000e+03
            TMR2P3  2.7805000e+05  -2.5250000e+01   6.5013000e+03
MONTE  CARLO RUN          3
                    ⋮
```

Note that the nominal trajectory values are presented in the first Monte Carlo case, so that the burst points of the timer fuzes all have the nominal altitudes. Also the crossrange components of the nominal burst points are not zero because of the spin of the earth. For the first off-nominal case, the downrange and crossrange components are different mainly due to the pre-reentry errors. The altitudes for the timer burst points are not the nomimal values because of the timer inaccuracies and the pre-reentry errors.

4.2 G-Started Timer Fuze

The g-started timer is a timer that starts its countdown after an accelerometer has reached a certain g level. So there are two sources of instrumentation errors as well as

AMEER name	G-started timer variable	AMEER name	G-started timer variable	AMEER name	G-started timer variable
GTMVL1	gt_{out_1}	GCLPPM(1)	ϵ_{g_1}	GTACL1	g_{out_1}
GTMVL2	gt_{out_2}	GCLPPM(2)	ϵ_{g_2}	GTACL2	g_{out_2}
GTMVL3	gt_{out_3}	GCLPPM(3)	ϵ_{g_3}	GTACL3	g_{out_3}
GTMVL4	gt_{out_4}	GCLPPM(4)	ϵ_{g_4}	GTACL4	g_{out_4}
GTMVL5	gt_{out_5}	GCLPPM(5)	ϵ_{g_5}	GTACL5	g_{out_5}
GTBIAS(I)	B_i	GTSENS(I)	S_i		

Table 7. AMEER Names for the G-Started Timer Fuze Variables

an intermediate stage to model. Again, up to five g-started timers can be modeled, each with up to ten presets. If a_i is the output from an AMEER accelerometer, S_i is the sensitivity of the accelerometer, B_i is the bias, and g_{out_i} is the accelerometer output for g-started timer i, then the accelerometer output (in G's) is modeled as:

$$g_{out_i} = S_i \times a_i + B_i \tag{34}$$

Once the correct g level is reached, the time is recorded; and the timer model is implemented. The g-started timer instrumentation errors are modeled as in the timer model except that Δt_{ref_i} is not modeled and t_{g_i} represents the time at which the accelerometer staged.

$$gt_{err_i} = 1{,}000{,}000 \times \epsilon_{g_i} \tag{35}$$

$$gt_{out_i} = (1 + gt_{err_i})(t - t_{g_i}) \tag{36}$$

The AMEER names for the accelerometer and timer variables just described are given in Table 7. The AMEER variable $NGTIMR$ is used to specify the number of g-started timers to be investigated. The AMEER array $GLEVEL(I)$ is used to store the g level on which g-started timer I is to stage, and the array $INGTAC(I)$ is used to store the index of which AMEER accelerometer g-started timer I uses. Though the fuze model allows for up to nine different AMEER accelerometers to be modeled, in most cases only one accelerometer will be used. Therefore, the value of the index will be 1. Even with only one accelerometer modeled, each g-started timer fuze can still stage on a different g level, and off-nominal fuzes will have a different set of instrumentation errors for each trajectory. The AMEER array $PRGTVL(I, J)$ is used to describe preset J for g-started timer I, and the array $NGPRST(I)$ is used to specify the number of presets for g-started timer I.

To illustrate the proper way to set up the g-started timer, the previous example will be expanded to include three g-started timer fuzes. The first one will start the

timer countdown at a g level of -10 g's, the second at -20 g's, and the third at -25 g's. The first two will use the same AMEER accelerometer model, and the third will use a second model. All are to fuze at 2000 and 3000 meters above the target. The one sigma sensitivity for the first accelerometer is 15 %, and 20 % for the second. All the fuzes have timer accuracies of 5000 ppm. A preliminary trajectory has shown the difference between the time the first accelerometer staged at -10 g's and the time to 8000 meters altitude (3000 meters above target) is 30 seconds and to 7000 meters is 31 seconds. For the second g-started timer, the differences are 25 and 26 seconds, and for the third, they are 20 and 21 seconds. As before, all the initial conditions, mass properties and aerodynamics are assumed to have already been described on *tp*5. In addition, the two accelerometers are assumed to have already been described. Only the lines that need to be changed from the previous example will be shown.

```
NUMFUZ 15 // total number of presets for all fuzes
NGTIMR 3 // number of g-started timer fuzes
⋮
NGPRST(1) 2 // number of presets for g-started timer 1
NTPRST(2) 2 // number of presets for g-started timer 2
NTPRST(3) 2 // number of presets for g-started timer 3
GLEVEL(1) -10.0 //g level for g-started timer 1
GLEVEL(2) -20.0 //g level for g-started timer 2
GLEVEL(3) -25.0 //g level for g-started timer 3
INGTAC(1) 1 //accelerometer index for g-started timer 1
INGTAC(2) 1 //accelerometer index for g-started timer 2
INGTAC(3) 2 //accelerometer index for g-started timer 3
*MONTE GCLPPM(1) NORMAL 0.0 5000.0
*MONTE GCLPPM(2) NORMAL 0.0 5000.0
*MONTE GCLPPM(3) NORMAL 0.0 5000.0
*MONTE GTSENS(1) NORMAL 1.0 0.15
*MONTE GTSENS(2) NORMAL 1.0 0.15
*MONTE GTSENS(3) NORMAL 1.0 0.20
PRGTVL(1,1) 30.0 // g-started timer 1 preset 1
PRGTVL(1,2) 31.0 // g-started timer 1 preset 2
PRGTVL(2,1) 25.0 // g-started timer 2 preset 1
PRGTVL(2,2) 26.0 // g-started timer 2 preset 2
PRGTVL(3,1) 20.0 // g-started timer 3 preset 1
PRGTVL(3,2) 21.0 // g-started timer 3 preset 2
⋮
NAMFUZ(10) $GTM1P1$ //g-started timer 1 preset 1
NAMFUZ(11) $GTM1P2$ //g-started timer 1 preset 2
NAMFUZ(12) $GTM2P1$ //g-started timer 2 preset 1
NAMFUZ(13) $GTM2P2$ //g-started timer 2 preset 2
```

```
NAMFUZ(14) $GTM3P1$ //g-started timer 3 preset 1
NAMFUZ(15) $GTM3P2$ //g-started timer 3 preset 2
NAMSTG(1) $GTACL1$ //stage on g level for first g-started timer
VALSTG(1) -10.0
NAMSTG(2) $GTACL2$ //stage on g level for second g-started timer
VALSTG(2) -20.0
NAMSTG(3) $GTACL3$ //stage on g level for third g-started timer
VALSTG(3) -25.0
*RUN
*RUN
*RUN
NAMSTG(1) $HGD8M$ //stage on first altitude
VALSTG(1) 9000.0
NAMSTG(2) $TIMVL1$ //stage on first preset for timer 1
VALSTG(2) 40.0
NAMSTG(3) $TIMVL2$ //stage on first preset for timer 2
VALSTG(3) 40.0
NAMSTG(4) $GTMVL1$ //stage on first preset for g-started timer 2
VALSTG(4) 20.0
NAMSTG(5) $GTMVL2$ //stage on first preset for g-started timer 2
VALSTG(5) 25.0
NAMSTG(3) $GTMVL3$ //stage on first preset for g-started timer 2
VALSTG(3) 30.0
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*END
```

Note that the mean for the sensitivities for all three accelerometer models is 1.0, and that the bias was not modeled. The first three $*RUN$ lines are used to stage on the proper

AMEER name	FBIA variable	AMEER name	FBIA variable	AMEER name	FBIA variable
FBVEL1	ΔV_1	FBSENS(1)	S_{f_1}	FBSRT1	f_{out_1}
FBVEL2	ΔV_2	FBSENS(2)	S_{f_2}	FBSRT2	f_{out_2}
FBVEL3	ΔV_3	FBSENS(3)	S_{f_3}	FBSRT3	f_{out_3}
FBVEL4	ΔV_4	FBSENS(4)	S_{f_4}	FBSRT4	f_{out_4}
FBVEL5	ΔV_5	FBSENS(5)	S_{f_5}	FBSRT5	f_{out_5}
FBBIAS(I)	B_{f_i}	FBGLVL(I)	g_{f_i}		

Table 8. AMEER Names for the FBIA Fuze Variables

acceleration for each g-started fuze. The final 15 $*RUN$ lines are used to determine the 15 burst points per trajectory (for all fuzes) necessary for the postprocessor. Since the output file *tape*25 is the same as before except for the addition of the six g-started timer burst points per trajectory, it is shown for this example.

4.3 Force-Balance Integrating Accelerometer (FBIA) Fuze

The FBIA (or velocity decrement) fuze uses a force-balance integrating accelerometer to measure velocity decrement. The fuze provides an output when the velocity decrement reaches a preset value. The AMEER FBIA model numerically integrates the output from an accelerometer to compute the decremental velocity. As with the g-started timer, the modeled accelerometer output must reach a certain level (g_{f_i}) before AMEER starts computing the velocity. The FBIA accelerometer output (f_{out_i}) is modeled similarly to the g-started timer accelerometer (with sensitivity S_{f_i}, bias B_{f_i}, and a_i as before).

$$f_{out_i} = S_{f_i} \times a_i + B_{f_i} \tag{37}$$

The velocity decrement (ΔV_i) is modeled as:

$$\Delta V_i = \int_{t_{ref}}^{t} (G)(f_{out_i})dt \tag{38}$$

where t_{ref} is the time at which the FBIA started computing the velocity and G is the gravitational acceleration.

The AMEER names for the FBIA fuze variables are contained in Table 8. The AMEER variable $NFBIA$ specifies the number FBIA fuzes; $INFBAC(I)$ specifies the accelerometer index; $PRFBVL(I,J)$ specifies the presets; and $NFPRST(I)$ specifies the number of presets for each FBIA fuze.

The example presented for the two previous fuze descriptions is continued for the FBIA fuze. One FBIA fuze is investigated with nominal burst points of 2000 and 3000

meters above the target. The accelerometer has a one sigma sensitivity value of 15 %; the bias has a uniform distribution of ± 0.1 g about zero; and the AMEER accelerometer model is the same as for the first g-started timer. The accelerometer must reach a level of 2.0 g's before the velocity decrement computation is begun. The nominal trajectory has shown that the velocity decrement to 3000 m above the target is 5500 m/s and to 2000 m above the target is 5520 m/s . Again only those lines that are different from the previous example or added will be shown.

```
NUMFUZ 17 // total number of presets for all fuzes
NFBIA 1 // number of FBIA fuzes
⋮
NFPRST(1) 2 // number of presets for FBIA 1
FBGLVL(1) -2.0 //g level for FBIA 1
INFBAC(1) 1 //accelerometer index for FBIA 1
*MONTE FBSENS(1) NORMAL 1.0 0.15
*MONTE FBBIAS(1) UNIFORM 0.0 0.1
PRFBVL(1,1) -5500.0 // FBIA 1 preset 1
PRFBVL(1,2) -5520.0 // FBIA 1 preset 2
⋮
NAMFUZ(16) $FBIA1$ //FBIA 1 preset 1
NAMFUZ(17) $FBIA2$ //FBIA 1 preset 2
NAMSTG(1) $FBSRT1$ //stage on g level for first FBIA
VALSTG(1) -2.0
*RUN
NAMSTG(1) $GTACL1$ //stage on g level for first g-started timer
VALSTG(1) -10.0
NAMSTG(2) $GTACL2$ //stage on g level for second g-started timer
VALSTG(2) -20.0
NAMSTG(3) $GTACL3$ //stage on g level for third g-started timer
VALSTG(3) -25.0
*RUN
*RUN
*RUN
NAMSTG(1) $HGD8M$ //stage on first altitude
VALSTG(1) 9000.0
NAMSTG(2) $TIMVL1$ //stage on first preset for timer 1
VALSTG(2) 40.0
NAMSTG(3) $TIMVL2$ //stage on first preset for timer 2
VALSTG(3) 40.0
NAMSTG(4) $GTMVL1$ //stage on first preset for g-started timer 2
VALSTG(4) 20.0
NAMSTG(5) $GTMVL2$ //stage on first preset for g-started timer 2
```

```
VALSTG(5) 25.0
NAMSTG(6) $GTMVL3$ //stage on first preset for g-started timer 3
VALSTG(6) 30.0
NAMSTG(7) $FBVEL1$ //stage on first preset for FBIA 1
VALSTG(7) -5500.0
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*END
```

Note that the first stage finds the 2 g level that begins the FBIA velocity decrement calculations. Then it is followed by the three stages to find the acceleration levels necessary for the g-started timer fuzes. The final 17 stages then are to find the 17 burst points per trajectory for all the fuzes.

4.4 Path Length Fuze

Since it would be very expensive to have the three-axis platform necessary for a fuze to measure the true path length traveled by a reentry body, most fuzes use an approximation based on a single axis accelerometer system. The approximation uses a term that is the double integral of the longitudinal deceleration (or drag), a term that is a velocity constant (C_{V_i}) multiplied by time, and a term that is a gravity constant (G_{G_i}) multiplied by time squared. The initial velocity is traditionally used for C_{V_i}, and an average over the trajectory for $\frac{1}{2}G\sin\gamma_{GD}$ is used for C_{G_i}. This represents one half the average contribution of gravity along the longitudinal axis of the vehicle. Since the path length fuze uses both an accelerometer and a timer, the previously described instrumentation errors for both are included in the AMEER path length fuze model. The

AMEER name	Path Length variable	AMEER name	Path Length variable	AMEER name	Path Length variable
PLEN1	PL_1	PCLPPM(1)	ϵ_{p_1}	PLTOP(1)	Δpt_{ref_1}
PLEN2	PL_2	PCLPPM(2)	ϵ_{p_2}	PLTOP(2)	Δpt_{ref_2}
PLEN3	PL_3	PCLPPM(3)	ϵ_{p_3}	PLTOP(3)	Δpt_{ref_3}
PLEN4	PL_4	PCLPPM(4)	ϵ_{p_4}	PLTOP(4)	Δpt_{ref_4}
PLEN5	PL_5	PCLPPM(5)	ϵ_{p_5}	PLTOP(5)	Δpt_{ref_5}
PLBIAS(I)	B_{p_i}	PLSENS(I)	S_{p_i}	PLVCNS(I)	C_{V_i}
PLGCNS(I)	C_{G_i}				

Table 9. AMEER Names for the Path Length Fuze Variables

path length (PL_i) is modeled in AMEER as:

$$PL_i = \int_0^t \int_0^t (G)(p_{out_i})dt^2 + C_{V_i} \times pt_{out_i} + C_{G_i} \times pt_{out_i}^2 \tag{39}$$

with G being the gravitational acceleration. The timer output pt_{out_i} and the accelerometer output p_{out_i} are modeled as:

$$pt_{err_i} = 1{,}000{,}000 \times \epsilon_{p_i} \tag{40}$$

$$pt_{out_i} = (pt_{err_i})(\Delta pt_{ref_i}) + (1 + pt_{err_i})t \tag{41}$$

$$p_{out_i} = S_{p_i} \times a_i + B_{p_i} \tag{42}$$

The AMEER names for the path length fuze variables are shown in Table 9. The AMEER variable $NPLEN$ is used to specify the number of path length fuzes; the AMEER array $INPLAC(I)$ is used to specify the accelerometer index for each fuze; the array $PRPLVL(I,J)$ is used to specify the preset for each fuze; and the array $NPPRST(I)$ is used to specify the number of presets per fuze.

The example will once more be expanded to include a path length fuze calculation. Let's suppose one path length fuze will be investigated. It will have nominal burst points of 3000 and 2000 meters above the target. The AMEER accelerometer used is the same as for the third g-started timer, but with a sensitivity of 3 %. The timer has an accuracy of 1000 ppm; and Δpt_{ref} is 800 seconds. The initial velocity is 5000 m/s. The first preliminary trajectory showed that the average value for $\frac{1}{2}G\sin\gamma_{GD}$ was 15 m/s^2. The second preliminary trajectory (with the path length fuze modeled) showed that the path lengths to the burst points were 300 km and 302 km. Again, only those lines that are used to model the path length fuze or are different from the previous example will be shown.

```
NUMFUZ 19 // total number of presets for all fuzes
NPLEN 1 // number of path length fuzes
```

```
⋮
NPPRST(1) 2 // number of presets for path length 1
PLGCNS(1) 15.0 //G constant for path length 1
PLVCNS(1) 5000.0 //V constant for path length 1
PLTOP(1) 800.0 //Delta t_ref for path length 1
INPLAC(1) 2 //accelerometer index for path length 1
*MONTE PLSENS(1) NORMAL 1.0 0.03
*MONTE PLPPM(1) NORMAL 0.0 1000.0
PRPLVL(1,1) 300000.0 // path length 1 preset 1
PRPLVL(1,2) 302000.0 // path length 1 preset 2
⋮
NAMFUZ(18) $PLEN1$ //path length 1 preset 1
NAMFUZ(19) $PLEN2$ //path length 1 preset 2
⋮
NAMSTG(8) $PLEN1$ //stage on first preset for path length 1
VALSTG(8) 600000.0
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*END
```

Note, that apart from the lines used to describe the path length parameters, the only lines that are different from the previous example are the $NUMFUZ$ line and the two extra $*RUN$ (to total 19) lines at the end.

AMEER name	Radar variable	AMEER name	Radar variable	AMEER name	Radar variable
RDALT1	h_{out_1}	RDBIAS(1)	B_{R_1}	RDRACC(1)	S_{R_1}
RDALT2	h_{out_2}	RDBIAS(2)	B_{R_2}	RDRACC(2)	S_{R_2}
RDALT3	h_{out_3}	RDBIAS(3)	B_{R_3}	RDRACC(3)	S_{R_3}
RDALT4	h_{out_4}	RDBIAS(4)	B_{R_4}	RDRACC(4)	S_{R_4}
RDALT5	h_{out_5}	RDBIAS(5)	B_{R_5}	RDRACC(5)	S_{R_5}

Table 10. AMEER Names for the Radar Fuze Variables

4.5 Radar Fuze

The same radar modeling is used for both the radar and RUPL fuze models. The generic radar output h_{R_i} is modeled with radar sensitivity S_{R_i} and bias B_{R_i} as:

$$h_{R_i} = S_{R_i}(h_{gd} - h_{target}) + B_{R_i} \tag{43}$$

The modeled radar output for the radar fuze is h_{out_i}. The AMEER names for the radar fuze variables are listed in Table 10. For the radar fuze model, the AMEER variable $NRADAR$ is used to specify the number of radar fuzes; $PRRDVL(I, J)$ is used to specify the presets; and $NRPRST(I)$ is used to specify the number of presets. The example for the radar fuze will be discussed in conjunction with the RUPL fuze example.

4.6 Radar Updated Path Length (RUPL) Fuze

The RUPL fuze is just a path length fuze that makes a correction to the path length computation at a reference altitude. The correction to the path length compensates for the fact that the uncorrected path length may fall long (or short) of the nominal burst point altitude. The concept is illustrated in Figure 3.

The ellipse at the top of the figure (not to scale) represents the distribution of reentry points as determined by the pre-reentry covariance matrix. The trajectory starting in the middle of the ellipse and ending at burst point 2 is the nominal trajectory. The trajectory starting at point A and ending at 3' is a typical uncorrected "high" trajectory, and the trajectory begining at B and ending at 1' is a typical uncorrected "low" trajectory. As they stand, the burst points 1' and 3' are far from the optimal altitude; however, means exist to move the burst altitudes to points 1 and 3. Suppose that the fuze can make a comparison of the computed path length to the nominal path length at a reference altitude. The fuze would use a radar to determine the reference altitude. It would find that the path length for trajectory A was longer than the nominal and that the computed path length for B was shorter. If it subtracts some fixed amount of path length from the computed path length for A, then the trajectory could continue until the more optimal

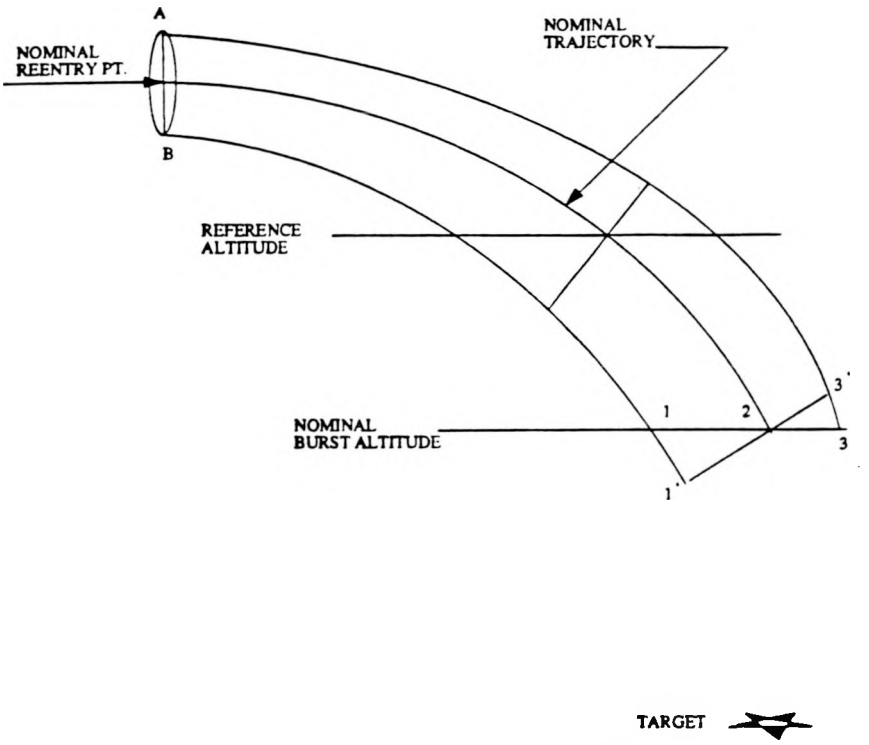

Figure 3. RUPL Fuze

burst point 3. Similarly, if the fuze adds some fixed amount of path to the computed path length for trajectory B, the burst point could be moved to 1.

The RUPL fuze model in AMEER allows for two such "radar updates" in its model. At two reference altitudes the measured path length (PL_{rupl_i}) is corrected by an amount equal to the difference between the measured path length and the the nominal path length times a gain factor($G1_i$ and $G2_i$). For the first and second RUPL altitudes ($h1_{rupl_i}$ and $h2_{rupl_i}$), the corrections ($\Delta PL1_i$ and $\Delta PL2_i$) are given by:

$$\Delta PL1_i = G1_i(PL_{rupl_i} - PL1_{nom_i}) \quad (44)$$

$$\Delta PL2_i = G2_i(PL_{rupl_i} - PL2_{nom_i}) \quad (45)$$

The path length calculation is modeled as before.

$$rt_{err_i} = 1{,}000{,}000 \times \epsilon_{r_i} \quad (46)$$

$$rt_{out_i} = (rt_{err_i})(\Delta rt_{ref_i}) + (1 + rt_{err_i})t \quad (47)$$

$$r_{out_i} = S_{r_i} \times a_i + B_{r_i} \quad (48)$$

$$PL_{rupl_i} = \int_0^t \int_0^t (G)(r_{out_i})dt^2 + CR_{V_i} \times rt_{out_i} + CR_{G_i} \times rt_{out_i}^2 \quad (49)$$

The AMEER names for the path length and update related variables for the RUPL fuze are listed in Table 11. Additionally, the AMEER variable $NRUPL$ specifies the

AMEER name	RUPL variable	AMEER name	RUPL variable	AMEER name	RUPL variable
SMSR1	PL_{rupl_1}	HUPDT1(1)	$h1_{rupl_1}$	HUPDT2(1)	$h2_{rupl_1}$
SMSR2	PL_{rupl_2}	HUPDT1(2)	$h1_{rupl_2}$	HUPDT2(2)	$h2_{rupl_2}$
SMSR3	PL_{rupl_3}	HUPDT1(3)	$h1_{rupl_3}$	HUPDT2(3)	$h2_{rupl_3}$
SMSR4	PL_{rupl_4}	HUPDT1(4)	$h1_{rupl_4}$	HUPDT2(4)	$h2_{rupl_4}$
SMSR5	PL_{rupl_5}	HUPDT1(5)	$h1_{rupl_5}$	HUPDT2(5)	$h2_{rupl_5}$
RUABIS(I)	B_{r_i}	RUSENS(I)	S_{r_i}	RUVCNS(I)	CR_{V_i}
RUGCNS(I)	CR_{G_i}	UCLPPM(I)	ϵ_{r_i}	RUTOP(I)	$Delta_{r_i}$
RGAIN1(I)	$G1_i$	RGAIN2(I)	$G2_i$	SRUPL1(I)	$PL1_{nom_i}$
SRUPL2(I)	$PL2_{nom_i}$				

Table 11. AMEER Names for the RUPL Path Length Related Variables

AMEER name	RUPL variable	AMEER name	RUPL variable	AMEER name	RUPL variable
RUPL11	$h1_{out_1}$	RUPL21	$h2_{out_1}$	RURACC(1)	S_{R_1}
RUPL12	$h1_{out_2}$	RUPL22	$h2_{out_2}$	RURACC(2)	S_{R_2}
RUPL13	$h1_{out_3}$	RUPL23	$h2_{out_3}$	RURACC(3)	S_{R_3}
RUPL14	$h1_{out_4}$	RUPL24	$h2_{out_4}$	RURACC(4)	S_{R_4}
RUPL15	$h1_{out_5}$	RUPL25	$h2_{out_5}$	RURACC(5)	S_{R_5}
RURBIS(I)	B_{R_i}				

Table 12. AMEER Names for the RUPL Radar Related Variables

number of RUPL fuzes; $INRUAC(I)$ specifies the accelerometer index; $PRRUVL(I,J)$ specifies the presets; and $NUPRST(I)$ specifies the number of presets. The RUPL fuze model must also include a provision for modeling the radar instrumentation errors as illustrated in Equation 43. AMEER has $h1_{out_i}$ being the modeled radar output for the first RUPL update altitude and $h2_{out_i}$ being the modeled output for the second RUPL update altitude. The AMEER names for radar related variables of the RUPL fuze model are listed in Table 12.

To complete the example, two RUPL fuzes each with two presets and a radar fuze with four presets will be added. Each RUPL fuze will update at 7000 and 5000 meters above the target with a radar sensitivity of 10%, and the burst point altitudes are 3000 and 2000 meters above the target. For both RUPL fuzes, the clock accuracy is 1000 ppm; the difference between fiducial and reentry times is 800 seconds; the accelerometer index is two; and the accelerometer has a sensitivity of 2 %. The radar fuze also has a one sigma

sensitivity 10%. Preliminary trajectories have shown the velocity and gravity constants required for the path length calculations are 5000 m/s and 15 m/s^2. The nominal path lengths to the two RUPL updates are 295 and 297.5 km and to the two burst points are 300 and 302 km. The first RUPL fuze has a gain of 0.50 and the second has a gain of 0.75. The four nominal altitudes for the radar fuze are 500, 1000, 1500, and 2000 m above the target. Only the lines that are different from or in addition to the previous example will be shown.

```
NUMFUZ 27 // total number of presets for all fuzes
NRUPL 2 // number of RUPL fuzes
NRADAR 1 // number of radar fuzes
⋮
NUPRST(1) 2 // number of presets for RUPL 1
NUPRST(2) 2 // number of presets for RUPL 2
NRPRST(1) 4 // number of presets for radar 1
RUGCNS(1) 15.0 //G constant for RUPL 1
RUGCNS(2) 15.0 //G constant for RUPL 2
RUVCNS(1) 5000.0 //V constant for RUPL 1
RUVCNS(2) 5000.0 //V constant for RUPL 2
RUTOP(1) 800.0 //Delta t_ref for RUPL 1
RUTOP(2) 800.0 //Delta t_ref for RUPL 2
INRUAC(1) 2 //accelerometer index for RUPL 1
INRUAC(2) 2 //accelerometer index for RUPL 2
HUPDT1(1) 7000.0 //first update altitude for RUPL 1
HUPDT2(1) 5000.0 //second update altitude for RUPL 1
SRUPL1(1) 295000.0 //nominal PL at first update altitude for RUPL 1
SRUPL2(1) 297500.0 //nominal PL at second update altitude for RUPL 1
RGAIN1(1) .50 //gain at first update altitude for RUPL 1
RGAIN2(1) .50 //gain at second update altitude for RUPL 1
HUPDT1(2) 7000.0 //first update altitude for RUPL 2
HUPDT2(2) 5000.0 //second update altitude for RUPL 2
SRUPL1(2) 295000.0 //nominal PL at first update altitude for RUPL 2
SRUPL2(2) 297500.0 //nominal PL at second update altitude for RUPL 2
RGAIN1(2) .75 //gain at first update altitude for RUPL 2
RGAIN2(2) .75 //gain at second update altitude for RUPL 2
*MONTE RUSENS(1) NORMAL 1.0 0.02
*MONTE RUSENS(2) NORMAL 1.0 0.02
*MONTE RUPPM(1) NORMAL 0.0 1000.0
*MONTE RUPPM(2) NORMAL 0.0 1000.0
*MONTE RURACC(1) NORMAL 1.0 0.10
*MONTE RURACC(2) NORMAL 1.0 0.10
*MONTE RDRACC(1) NORMAL 1.0 0.10
PRRUVL(1,1) 300000.0 // RUPL 1 preset 1
```

```
PRRUVL(1,2) 302000.0 // RUPL 1 preset 2
PRRUVL(2,1) 300000.0 // RUPL 2 preset 1
PRRUVL(2,2) 302000.0 // RUPL 2 preset 2
PRRDVL(1,1) 2000.0 // radar 1 preset 1
PRRDVL(1,2) 1500.0 // radar 1 preset 2
PRRDVL(1,3) 1000.0 // radar 1 preset 3
PRRDVL(1,4) 500.0 // radar 1 preset 4
⋮
NAMFUZ(20) $RUPL11$ //RUPL 1 preset 1
NAMFUZ(21) $RUPL12$ //RUPL 1 preset 2
NAMFUZ(22) $RUPL21$ //RUPL 2 preset 1
NAMFUZ(23) $RUPL22$ //RUPL 2 preset 2
NAMFUZ(24) $RADAR1$ //radar preset 1
NAMFUZ(25) $RADAR2$ //radar preset 2
NAMFUZ(26) $RADAR3$ //radar preset 3
NAMFUZ(27) $RADAR4$ //radar preset 4
NAMSTG(1) $FBSRT1$ //stage on g level for first FBIA
VALSTG(1) -2.0
*RUN
NAMSTG(1) $GTACL1$ //stage on g level for first g-started timer
VALSTG(1) -10.0
NAMSTG(2) $GTACL2$ //stage on g level for second g-started timer
VALSTG(2) -20.0
NAMSTG(3) $GTACL3$ //stage on g level for third g-started timer
VALSTG(3) -25.0
*RUN
*RUN
*RUN
NAMSTG(1) $RUPL11$ //stage on first RUPL 1 altitude
VALSTG(1) 12000.0
NAMSTG(1) $RUPL12$ //stage on first RUPL 2 altitude
VALSTG(1) 12000.0
*RUN
*RUN
NAMSTG(1) $HGD8M$ //stage on first altitude
VALSTG(1) 9000.0
NAMSTG(2) $TIMVL1$ //stage on first preset for timer 1
VALSTG(2) 40.0
NAMSTG(3) $TIMVL2$ //stage on first preset for timer 2
VALSTG(3) 40.0
NAMSTG(4) $GTMVL1$ //stage on first preset for g-started timer 2
VALSTG(4) 20.0
```

```
NAMSTG(5) $GTMVL2$ //stage on first preset for g-started timer 2
VALSTG(5) 25.0
NAMSTG(6) $GTMVL3$ //stage on first preset for g-started timer 3
VALSTG(6) 30.0
NAMSTG(7) $FBVEL1$ //stage on first preset for FBIA 1
VALSTG(7) -5500.0
NAMSTG(8) $PLEN1$ //stage on first preset for path length 1
VALSTG(8) 600000.0
NAMSTG(9) $SMSR1$ //stage on first preset for RUPL 1
VALSTG(9) 600000.0
NAMSTG(10) $SMSR2$ //stage on first preset for RUPL 2
VALSTG(10) 600000.0
NAMSTG(11) $RDALT2$ //stage on first preset for radar 1
VALSTG(11) 2000.0
NAMSTG(12) $RUPL21$ //stage on second RUPL 1 altitude
VALSTG(12) 10000.0
NAMSTG(13) $RUPL22$ //stage on second RUPL 2 altitude
VALSTG(13) 10000.0
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
*RUN
```

```
*RUN
*RUN
*RUN
*RUN
*END
```

Note that there are 29 final stages, 27 for the presets of all the fuzes and two for the second update altitude for the two RUPL fuzes. The second updates were included in the final set of fuzes since it is possible that, with some combination of errors, some of the fuzes could go off before the second update.

As the MCPRAM input file now stands, it would generate an error when trying to run the first (nominal) trajectory on AMEER. This is because on the nominal trajectory, the two stages for the first RUPL updates only require one stage since they are for the same nominal altitude. **The user must remove the first trajectory out of the AMEER input file to get AMEER to work.** Unfortunately, this is an anomaly that cannot be eliminated very easily so it was included in the example to warn future users. One other observation is that the 27 presets are not defined until late in the trajectory (after the first RUPL update). This keeps the computational cost down since the cost of a search for 27 stages by the numerical integrator is relatively expensive. For the first trajectory, the fuze output file (*tape*25) should look like:

```
MONTE  CARLO RUN          2
          HCTOFF  2.8518000e+05  -2.5700000e+01  2.0000000e+03
          HTRGET  2.8145000e+05  -2.5400000e+01  5.0000000e+04
            H6000  2.7930000e+05  -2.5300000e+01  6.0000000e+04
            H9000  2.9885000e+05  -2.4000000e+01  9.0000000e+03
          TMR1P1  2.7500000e+05  -2.5000000e+01  8.0100000e+03
          TMR1P2  2.7710000e+05  -2.5200000e+01  7.0110000e+03
          TMR2P1  2.7510000e+05  -2.5000000e+01  8.0090000e+03
          TMR2P2  2.7610000e+05  -2.5100000e+01  7.5110000e+03
          TMR2P3  2.7805000e+05  -2.5250000e+01  6.5013000e+03
          GTM1P1  2.7501400e+05  -2.5040000e+01  8.0120000e+03
          GTM1P2  2.7710300e+05  -2.5250000e+01  7.0150000e+03
          GTM2P1  2.7501400e+05  -2.5040000e+01  8.0126000e+03
          GTM2P2  2.7710300e+05  -2.5250000e+01  7.0157000e+03
          GTM3P1  2.7502400e+05  -2.5540000e+01  8.0140000e+03
          GTM3P2  2.7711300e+05  -2.5650000e+01  7.0170000e+03
            FBIA1  2.7501600e+05  -2.5060000e+01  8.0110000e+03
            FBIA2  2.7710500e+05  -2.5270000e+01  7.0145000e+03
            PLEN1  2.7502000e+05  -2.5160000e+01  7.9940000e+03
            PLEN2  2.7712100e+05  -2.5370000e+01  6.9965000e+03
```

```
                RUPL11  2.7500700e+05  -2.5180000e+01  8.0070000e+03
                RUPL12  2.7711600e+05  -2.5170000e+01  7.0065000e+03
                RUPL21  2.7500600e+05  -2.5280000e+01  8.0030000e+03
                RUPL22  2.7711500e+05  -2.5070000e+01  7.0035000e+03
                RADAR1  2.7700600e+05  -2.5080000e+01  7.2030000e+03
                RADAR2  2.7811500e+05  -2.5070000e+01  6.6535000e+03
                RADAR3  2.7900600e+05  -2.5060000e+01  6.1030000e+03
                RADAR4  2.7951500e+05  -2.5050000e+01  5.5435000e+03
MONTE  CARLO RUN              3
    :
```

This time the output for the first trajectory is the first off-nomimal set of burst points since the input for the nominal trajectory had been removed. The scoring program SCORES can take this information and compute a P_k based on actual guidance and fuzing errors along with other target and weapon characteristics. The codes can be used to evaluate fuzing philosophies for fuzes such as the g-started timer and the RUPL.

5 Conclusions

MCPRAM, a preprocessor for the AMEER 6-DOF trajectory code, uses Monte Carlo methods to create an AMEER input file to compute multiple trajectories. MCPRAM allows users to select which AMEER input variable to investigate and the type of probability distribution for the variable to have (normal, uniform, or Rayleigh). MCPRAM uses random number generators from the IMSL library and user-supplied information to generate a series of values for each variable. MCPRAM also has the ability to use a covariance analysis method to generate random values for sets of correlated parameters. For each of seven sets, the eigenvalues and eigenvector matrix of a user-supplied covariance matrix and a normally distributed random number generator are used to generate the correlated random values. The predefined sets of correlated variables generate pre-reentry errors and wind, axial force, and atmospheric models. The new AMEER variables that were generated by the pre-reentry and axial force covariance matrices were presented.

In conjunction with MCPRAM, AMEER was modified to include provisions for six types of fuze models. The fuze types include timer, g-started timer, FBIA, path length, radar, and RUPL. As each fuze was dicussed, an example describing how to use the fuze model was given. Five versions (or less) of each fuze with up to ten presets per fuze can be used. The AMEER fuze model generates an output file of burst point locations. The burst point information is given in downrange, crossrange, and altitude components based on a nomimal trajectory flight path angle. A separate scoring program (SCORES) must use the output file to evaluate weapon system effectiveness. Although a contact fuze model was not included in the discussions, there is enough information in the AMEER output file for SCORES to evaluate its effectiveness.

References

1. Meyer, Eugene J., *A User's Manual for the AMEER Flight Path Trajectory Simulation Code*, **SAND80-2056 Revised**, December 1984.

2. LaFarge, Robert A., *A Description of the Reference Frames and the Coordinate Transformations Used in AMEER*, **SAND88-0243**, March 1988.

3. Binder Kurt, Editor, *Monte Carlo Methods in Statistical Physics*, Springer-Verlag, Berlin : Heidelburg : New York : Tokyo, 1986.

4. Rice, John R., *Numerical Methods, Software, and Analysis: IMSL Reference Edition*, McGraw-Hill Inc., New York : St Louis : San Francisco : Toronto : London Sydney, 1983.

5. Papoulis, Athanasios, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill Inc., New York : St Louis : San Francisco : Toronto : London Sydney, 1965.

6. Liebelt, Paul B., *An Introduction to Optimal Control*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts : Menlo Park, California : London : Don Mills, Ontario, 1967.

7. Ralston, Anthony, *A First Cource in Numerical Analysis*, McGraw-Hill Inc., New York : St Louis : San Francisco : Toronto : London Sydney, 1965.

Distribution:

Commander, U. S. Army
ARDEC
attn: E. F. Brown
SMCAR LAET-A
Picatinny, New Jersey 07806-5000

J. L. Forkois
Kaman Sciences Corporation
P. O. Box 7463
Colorado Springs, Colorado 80933-7463

Sandia Internal:

1500 E. H. Barsis
1510 E. H. Barsis, Actg.
1520 L. W. Davison
1530 J. R. Asay
1550 C. W. Peterson
1551 J. K. Cole
1551 C. E. Hailey
1551 A. E. Hodapp
1551 R. A. LaFarge (30)
1551 H. R. Spahr
1551 R. J. Weir
1551 W. P. Wolfe
1552 D. D. McBride
1553 W. H. Hermina
1554 D. P. Aeschliman
1555 C. W. Peterson, Actg.
1555 R. W. Greene
1555 T. M. Jordan
1555 D. L. Keese
1555 W. A. Millard
1555 D. E. Outka
1555 L. R. Rollstin
1556 W. L. Oberkampf
1556 G. A. Laguna
1556 A. R. Lopez
5151 G. A. Kinemond
5160 G. R. Otey
5166 R. C. Hartwig
5166 A. B. Cox
7222 T. J. Kerschen
8165 K. E. Carbiener
8171 R. N. Everett
9010 W. C. Hines
9014 M. L. Bunting
9014 J. M. Parvin (2)
9015 J. W. Purvis
9113 C. S. Lee (2)
9113 W. G. Kerschen
9144 W. E. Williamson
9144 S. A. Kerr
9144 J. L. McDowell
9144 B. A. Rainwater
9144 D. E. Salguero
3141 S. A. Landenberger (5)
3141-1 C. L. Ward (8)
3151 W. I. Klein (3)
8524 J. A. Wackerly